

11. Write the python program for map coloring to implement CSP.

```
class MapColoring:
```

```
    def __init__(self, graph, colors):
```

```
        self.graph = graph
```

```
        self.colors = colors
```

```
        self.assignment = {}
```

```
    def is_safe(self, node, color):
```

```
        for neighbor in self.graph[node]:
```

```
            if neighbor in self.assignment and self.assignment[neighbor] == color:
```

```
                return False
```

```
        return True
```

```
    def backtrack(self):
```

```
        if len(self.assignment) == len(self.graph):
```

```
            return self.assignment
```

```
        node = self.select_unassigned_node()
```

```
        for color in self.colors:
```

```
            if self.is_safe(node, color):
```

```
                self.assignment[node] = color
```

```
                result = self.backtrack()
```

```
                if result:
```

```
                    return result
```

```
                del self.assignment[node]
```

```
        return None
```

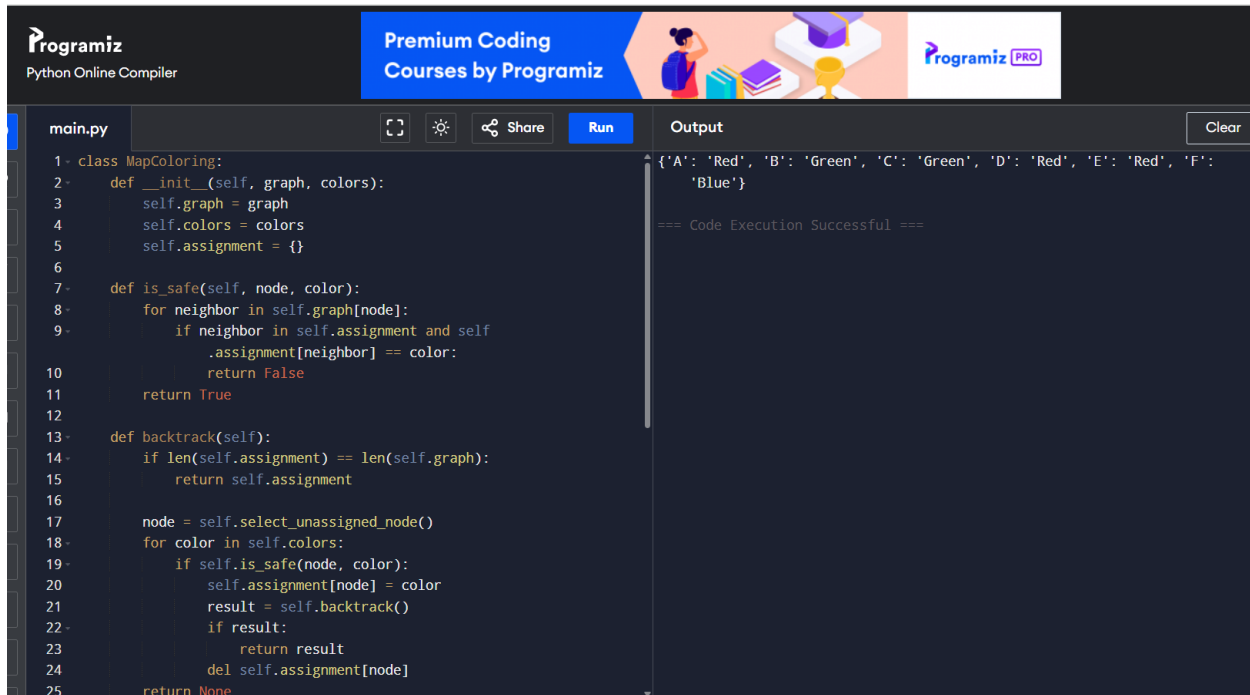
```
def select_unassigned_node(self):  
    for node in self.graph:  
        if node not in self.assignment:  
            return node  
    return None
```

```
def solve(self):  
    return self.backtrack()
```

Example usage

```
graph = {  
    'A': ['B', 'C'],  
    'B': ['A', 'D', 'E'],  
    'C': ['A', 'F'],  
    'D': ['B'],  
    'E': ['B', 'F'],  
    'F': ['C', 'E']  
}  
colors = ['Red', 'Green', 'Blue']
```

```
map_coloring = MapColoring(graph, colors)  
solution = map_coloring.solve()  
print(solution)
```



```
1- class MapColoring:
2-     def __init__(self, graph, colors):
3-         self.graph = graph
4-         self.colors = colors
5-         self.assignment = {}
6-
7-     def is_safe(self, node, color):
8-         for neighbor in self.graph[node]:
9-             if neighbor in self.assignment and self
               .assignment[neighbor] == color:
10-                 return False
11-         return True
12-
13-     def backtrack(self):
14-         if len(self.assignment) == len(self.graph):
15-             return self.assignment
16-
17-         node = self.select_unassigned_node()
18-         for color in self.colors:
19-             if self.is_safe(node, color):
20-                 self.assignment[node] = color
21-                 result = self.backtrack()
22-                 if result:
23-                     return result
24-                 del self.assignment[node]
25-         return None
```

Output

```
{'A': 'Red', 'B': 'Green', 'C': 'Green', 'D': 'Red', 'E': 'Red', 'F':
  'Blue'}

=== Code Execution Successful ===
```

12. Write the python program for Tic Tac Toc game.

```
def print_board(board):
```

```
    for row in board:
```

```
        print(" | ".join(row))
```

```
    print("-" * 9)
```

```
def check_winner(board):
```

```
    for row in board:
```

```
        if row.count(row[0]) == 3 and row[0] != " ":
```

```
            return True
```

```
    for col in range(3):
```

```
        if board[0][col] == board[1][col] == board[2][col] != " ":
```

```
            return True
```

```
    if board[0][0] == board[1][1] == board[2][2] != " " or board[0][2] == board[1][1] == board[2][0]
    != " ":
```

```
    return True
```

```
    return False
```

```
def tic_tac_toe():
```

```
    board = [[" " for _ in range(3)] for _ in range(3)]
```

```
    current_player = "X"
```

```
    for turn in range(9):
```

```
        print_board(board)
```

```
        row = int(input(f"Player {current_player}, enter your row (0-2): "))
```

```
        col = int(input(f"Player {current_player}, enter your column (0-2): "))
```

```
        if board[row][col] == " ":
```

```
            board[row][col] = current_player
```

```
            if check_winner(board):
```

```
                print_board(board)
```

```
                print(f"Player {current_player} wins!")
```

```
                return
```

```
            current_player = "O" if current_player == "X" else "X"
```

```
        else:
```

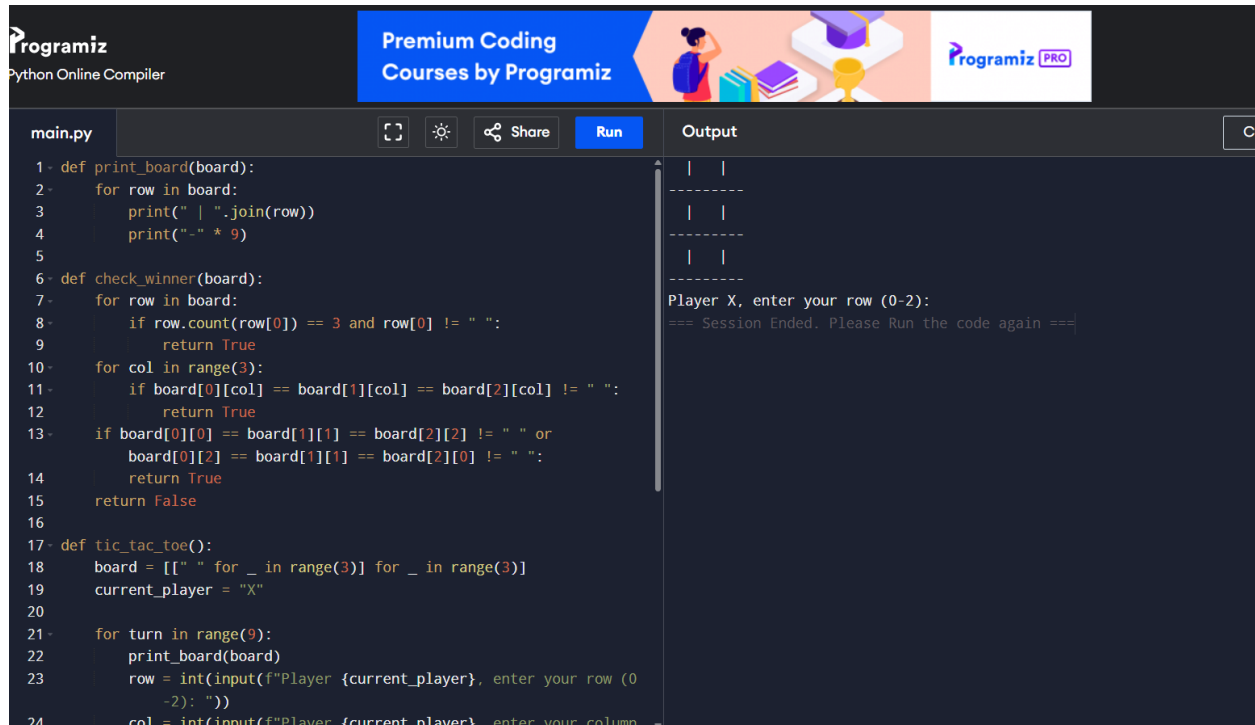
```
            print("Invalid move, try again.")
```

```
    print_board(board)
```

```
    print("It's a draw!")
```

```
if __name__ == "__main__":
```

tic_tac_toe()



The screenshot shows the Programiz Python Online Compiler interface. The top header includes the Programiz logo, the text "Python Online Compiler", and a blue banner for "Premium Coding Courses by Programiz". The main area is split into two panels. The left panel, titled "main.py", contains the following Python code:

```
1- def print_board(board):
2-     for row in board:
3-         print(" | ".join(row))
4-         print("-" * 9)
5-
6- def check_winner(board):
7-     for row in board:
8-         if row.count(row[0]) == 3 and row[0] != " ":
9-             return True
10-    for col in range(3):
11-        if board[0][col] == board[1][col] == board[2][col] != " ":
12-            return True
13-    if board[0][0] == board[1][1] == board[2][2] != " " or
14-       board[0][2] == board[1][1] == board[2][0] != " ":
15-        return True
16-    return False
17-
18- def tic_tac_toe():
19-     board = [" " for _ in range(3)] for _ in range(3)]
20-     current_player = "X"
21-
22-     for turn in range(9):
23-         print_board(board)
24-         row = int(input(f"Player {current_player}, enter your row (0-2): "))
25-         col = int(input(f"Player {current_player}, enter your column (0-2): "))
```

The right panel, titled "Output", shows the initial board state and the first prompt:

```

| | |
-----
| | |
-----
| | |
-----
Player X, enter your row (0-2):
=== Session Ended. Please Run the code again ===
```

13. write the python program to implement minimax algorithm for gaming.

```
def minimax(board, depth, is_maximizing):
```

```
    score = evaluate(board)
```

```
    if score == 10:
```

```
        return score - depth
```

```
    if score == -10:
```

```
        return score + depth
```

```
    if is_board_full(board):
```

```
        return 0
```

```
    if is_maximizing:
```

```
        best_value = float('-inf')
```

```
        for move in get_possible_moves(board):
```

```

board[move] = 'X' # Assume 'X' is the maximizing player

best_value = max(best_value, minimax(board, depth + 1, False))

board[move] = '' # Undo the move

return best_value

else:

    best_value = float('inf')

    for move in get_possible_moves(board):

        board[move] = 'O' # Assume 'O' is the minimizing player

        best_value = min(best_value, minimax(board, depth + 1, True))

        board[move] = '' # Undo the move

    return best_value

# Additional functions like evaluate, is_board_full, and get_possible_moves need to be defined.

```

The screenshot displays the Programiz Python Online Compiler interface. The left pane shows the code for a minimax algorithm. The right pane shows the output of the code execution.

Code (main.py):

```

1- def minimax(board, depth, is_maximizing):
2-     score = evaluate(board)
3-
4-     if score == 10:
5-         return score - depth
6-     if score == -10:
7-         return score + depth
8-     if is_board_full(board):
9-         return 0
10-
11-     if is_maximizing:
12-         best_value = float('-inf')
13-         for move in get_possible_moves(board):
14-             board[move] = 'X' # Assume 'X' is the maximizing player
15-             best_value = max(best_value, minimax(board, depth + 1,
16-                                                     False))
17-             board[move] = '' # Undo the move
18-         return best_value
19-     else:
20-         best_value = float('inf')
21-         for move in get_possible_moves(board):
22-             board[move] = 'O' # Assume 'O' is the minimizing player
23-             best_value = min(best_value, minimax(board, depth + 1,
24-                                                     True))
25-             board[move] = '' # Undo the move
26-         return best_value

```

Output:

```

Current board:
X|O|X
|O|
| |

AI recommends move at position: 5

Board after AI move:
X|O|X
|O|X
| |

=== Code Execution Successful ===

```

14. Write the python program to implement Apha and beta pruning algorithm for gaming .

```
def alpha_beta(node, depth, alpha, beta, maximizing_player):
```

```
    if depth == 0 or is_terminal(node):
```

```
    return evaluate(node)
```

```
if maximizing_player:
```

```
    max_eval = float('-inf')
```

```
    for child in get_children(node):
```

```
        eval = alpha_beta(child, depth - 1, alpha, beta, False)
```

```
        max_eval = max(max_eval, eval)
```

```
        alpha = max(alpha, eval)
```

```
        if beta <= alpha:
```

```
            break # Beta cut-off
```

```
    return max_eval
```

```
else:
```

```
    min_eval = float('inf')
```

```
    for child in get_children(node):
```

```
        eval = alpha_beta(child, depth - 1, alpha, beta, True)
```

```
        min_eval = min(min_eval, eval)
```

```
        beta = min(beta, eval)
```

```
        if beta <= alpha:
```

```
            break # Alpha cut-off
```

```
    return min_eval
```

```
# Example usage
```

```
# root = initial_game_state()
```

```
# best_value = alpha_beta(root, depth, float('-inf'), float('inf'), True)
```

The screenshot shows the Programiz Python Online Compiler interface. The editor contains a Python script for a Minimax algorithm. The script defines a recursive function `alpha_beta` that takes a node, depth, alpha, beta, and a maximizing player flag as arguments. It implements the minimax search logic with pruning (Alpha-Beta pruning). The output window shows the execution result: `P = Apha`, `m = beta`, and `=== Code Execution Successful ===`.

```
1- def alpha_beta(node, depth, alpha, beta, maximizing_player):
2-     if depth == 0 or is_terminal(node):
3-         return evaluate(node)
4-
5-     if maximizing_player:
6-         max_eval = float('-inf')
7-         for child in get_children(node):
8-             eval = alpha_beta(child, depth - 1, alpha, beta, False)
9-             max_eval = max(max_eval, eval)
10-            alpha = max(alpha, eval)
11-            if beta <= alpha:
12-                break # Beta cut-off
13-        return max_eval
14-     else:
15-         min_eval = float('inf')
16-         for child in get_children(node):
17-             eval = alpha_beta(child, depth - 1, alpha, beta, True)
18-             min_eval = min(min_eval, eval)
19-             beta = min(beta, eval)
20-             if beta <= alpha:
21-                 break # Alpha cut-off
22-        return min_eval
23-
24- # Example usage
25- # root = initial_game_state()
26- value = alpha_beta(root, depth, float('-inf'), float('inf'),
```

15. Write the python program to implement decision tree.

Import necessary libraries

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

import matplotlib.pyplot as plt

Load the Iris dataset

iris = load_iris()

X = iris.data

y = iris.target

Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


```
# Create a Decision Tree Classifier
```

```
clf = DecisionTreeClassifier()
```

```
# Train the model
```

```
clf.fit(X_train, y_train)
```

```
# Evaluate the model
```

```
accuracy = clf.score(X_test, y_test)
```

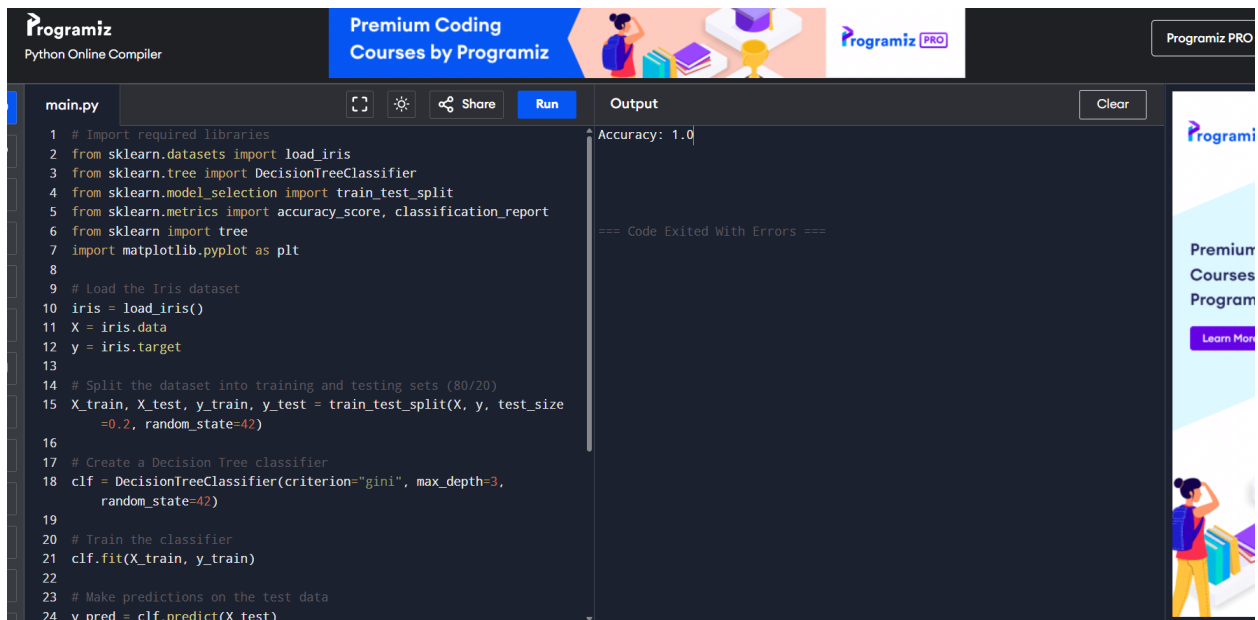
```
print(f"Accuracy: {accuracy:.2f}")
```

```
# Visualize the decision tree
```

```
plt.figure(figsize=(12,8))
```

```
tree.plot_tree(clf, filled=True)
```

```
plt.show()
```



The screenshot displays the Programiz Python Online Compiler interface. The top header includes the Programiz logo, the text "Python Online Compiler", a blue banner for "Premium Coding Courses by Programiz", and a "Programiz PRO" badge. The main area is divided into a code editor on the left and an output window on the right. The code editor shows a Python script for training and evaluating a Decision Tree Classifier on the Iris dataset. The output window shows the result "Accuracy: 1.0" and a message "=== Code Exited With Errors ===". A sidebar on the right promotes "Premium Courses Program" with a "Learn More" button.

```
main.py  [Icons]  Share  Run  Output  Clear
```

```
1 # Import required libraries
2 from sklearn.datasets import load_iris
3 from sklearn.tree import DecisionTreeClassifier
4 from sklearn.model_selection import train_test_split
5 from sklearn.metrics import accuracy_score, classification_report
6 from sklearn import tree
7 import matplotlib.pyplot as plt
8
9 # Load the Iris dataset
10 iris = load_iris()
11 X = iris.data
12 y = iris.target
13
14 # Split the dataset into training and testing sets (80/20)
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)
16
17 # Create a Decision Tree classifier
18 clf = DecisionTreeClassifier(criterion="gini", max_depth=3,
    random_state=42)
19
20 # Train the classifier
21 clf.fit(X_train, y_train)
22
23 # Make predictions on the test data
24 y_pred = clf.predict(X_test)
```

Accuracy: 1.0

=== Code Exited With Errors ===

Programiz PRO

Premium Courses Program

Learn More

16. Write the python program to implement feed forward neural network.

```
import numpy as np
```

```
class FeedForwardNN:
```

```
    def __init__(self, input_size, hidden_size, output_size):
```

```
        # Initialize weights
```

```
        self.weights_input_hidden = np.random.rand(input_size, hidden_size)
```

```
        self.weights_hidden_output = np.random.rand(hidden_size, output_size)
```

```
        self.bias_hidden = np.random.rand(hidden_size)
```

```
        self.bias_output = np.random.rand(output_size)
```

```
    def sigmoid(self, x):
```

```
        return 1 / (1 + np.exp(-x))
```

```
    def forward(self, x):
```

```
        # Input to hidden layer
```

```
        self.hidden_layer_input = np.dot(x, self.weights_input_hidden) + self.bias_hidden
```

```
        self.hidden_layer_output = self.sigmoid(self.hidden_layer_input)
```

```
        # Hidden to output layer
```

```
        self.output_layer_input = np.dot(self.hidden_layer_output, self.weights_hidden_output) +  
self.bias_output
```

```
        self.output = self.sigmoid(self.output_layer_input)
```

```
        return self.output
```

```
# Example usage
```

```
if __name__ == "__main__":
```

```

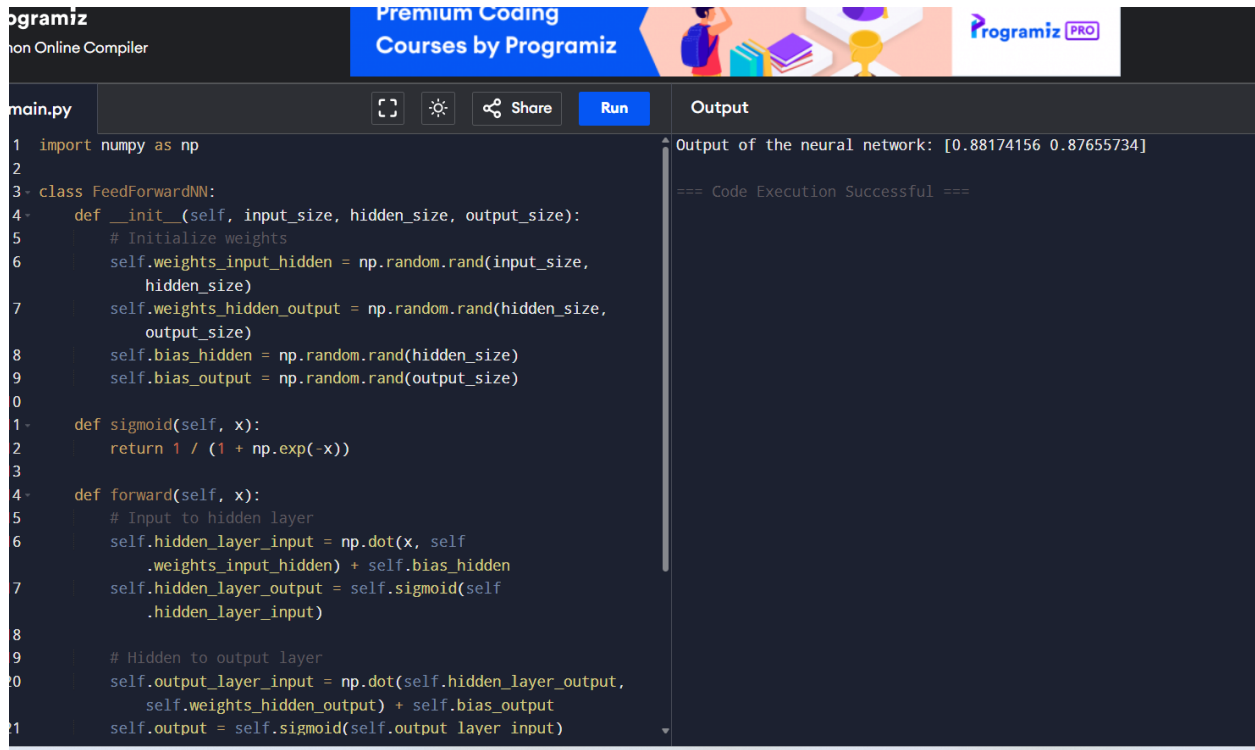
nn = FeedForwardNN(input_size=3, hidden_size=5, output_size=2)

input_data = np.array([0.1, 0.2, 0.3])

output = nn.forward(input_data)

print("Output of the neural network:", output)

```



The screenshot shows the Programiz Online Compiler interface. The code editor on the left contains a Python script for a FeedForwardNN class. The output panel on the right shows the result of the execution.

```

main.py
1 import numpy as np
2
3 class FeedForwardNN:
4     def __init__(self, input_size, hidden_size, output_size):
5         # Initialize weights
6         self.weights_input_hidden = np.random.rand(input_size,
7             hidden_size)
8         self.weights_hidden_output = np.random.rand(hidden_size,
9             output_size)
10        self.bias_hidden = np.random.rand(hidden_size)
11        self.bias_output = np.random.rand(output_size)
12
13    def sigmoid(self, x):
14        return 1 / (1 + np.exp(-x))
15
16    def forward(self, x):
17        # Input to hidden layer
18        self.hidden_layer_input = np.dot(x, self
19            .weights_input_hidden) + self.bias_hidden
20        self.hidden_layer_output = self.sigmoid(self
21            .hidden_layer_input)
22
23        # Hidden to output layer
24        self.output_layer_input = np.dot(self.hidden_layer_output,
25            self.weights_hidden_output) + self.bias_output
26        self.output = self.sigmoid(self.output layer input)

```

Output

```

Output of the neural network: [0.88174156 0.87655734]

=== Code Execution Successful ===

```

17. Write the python program to sum the integers from 1 to n.

```
def sum_integers(n):
```

```
    total = 0
```

```
    for i in range(1, n + 1):
```

```
        total += i
```

```
    return total
```

```
n = 10 # Example input
```

```
print(f"The sum of integers from 1 to {n} is: {sum_integers(n)}")
```



18. Write the prolog program for A DB WITH NAME,DOB.

```
import sqlite3
```

```
# Connect to the database (or create it if it doesn't exist)
```

```
conn = sqlite3.connect('people.db')
```

```
# Create a cursor object
```

```
cursor = conn.cursor()
```

```
# Create a table for storing names and DOB
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS person (
```

```
    id INTEGER PRIMARY KEY,
```

```
    name TEXT NOT NULL,
```

```
    dob DATE NOT NULL
```

```
)
```

```
""")
```

```
# Function to insert a new person into the database
```

```
def insert_person(name, dob):
```

```
    cursor.execute("""
```

```
    INSERT INTO person (name, dob) VALUES (?, ?)
```

```
    """, (name, dob))
```

```
    conn.commit()
```

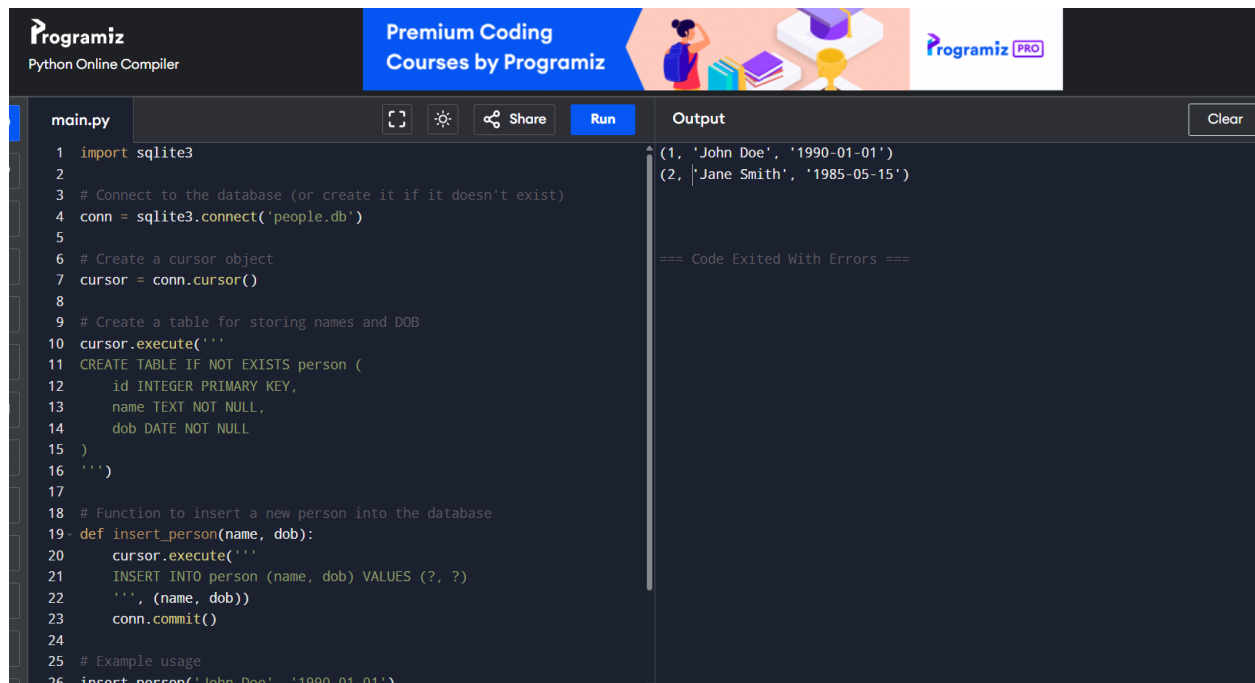
```
# Example usage
```

```
insert_person('John Doe', '1990-01-01')
```

```
insert_person('Jane Smith', '1985-05-15')
```

```
# Close the connection
```

```
conn.close()
```



The screenshot displays the Programiz Python Online Compiler interface. The top header includes the Programiz logo, the text "Python Online Compiler", a blue banner for "Premium Coding Courses by Programiz", and a "Programiz PRO" badge. The main area is divided into a code editor on the left and an output panel on the right. The code editor shows a file named "main.py" with the following Python code:

```
1 import sqlite3
2
3 # Connect to the database (or create it if it doesn't exist)
4 conn = sqlite3.connect('people.db')
5
6 # Create a cursor object
7 cursor = conn.cursor()
8
9 # Create a table for storing names and DOB
10 cursor.execute('''
11 CREATE TABLE IF NOT EXISTS person (
12     id INTEGER PRIMARY KEY,
13     name TEXT NOT NULL,
14     dob DATE NOT NULL
15 )
16 ''')
17
18 # Function to insert a new person into the database
19 def insert_person(name, dob):
20     cursor.execute('''
21     INSERT INTO person (name, dob) VALUES (?, ?)
22     ''', (name, dob))
23     conn.commit()
24
25 # Example usage
26 insert_person('John Doe', '1990-01-01')
```

The output panel on the right shows the execution results:

```
(1, 'John Doe', '1990-01-01')
(2, 'Jane Smith', '1985-05-15')
```

Below the output, it states "=== Code Exited With Errors ===".

19. Write the prolog program for STUDENT -TEACHER-SUB-CODE.

```
class Subject:
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
class Teacher:
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
        self.subjects = []
```

```
    def assign_subject(self, subject):
```

```
        self.subjects.append(subject)
```

```
class Student:
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
        self.subjects = []
```

```
    def enroll(self, subject):
```

```
        self.subjects.append(subject)
```

```
# Example usage
```

```
math = Subject("Mathematics")
```

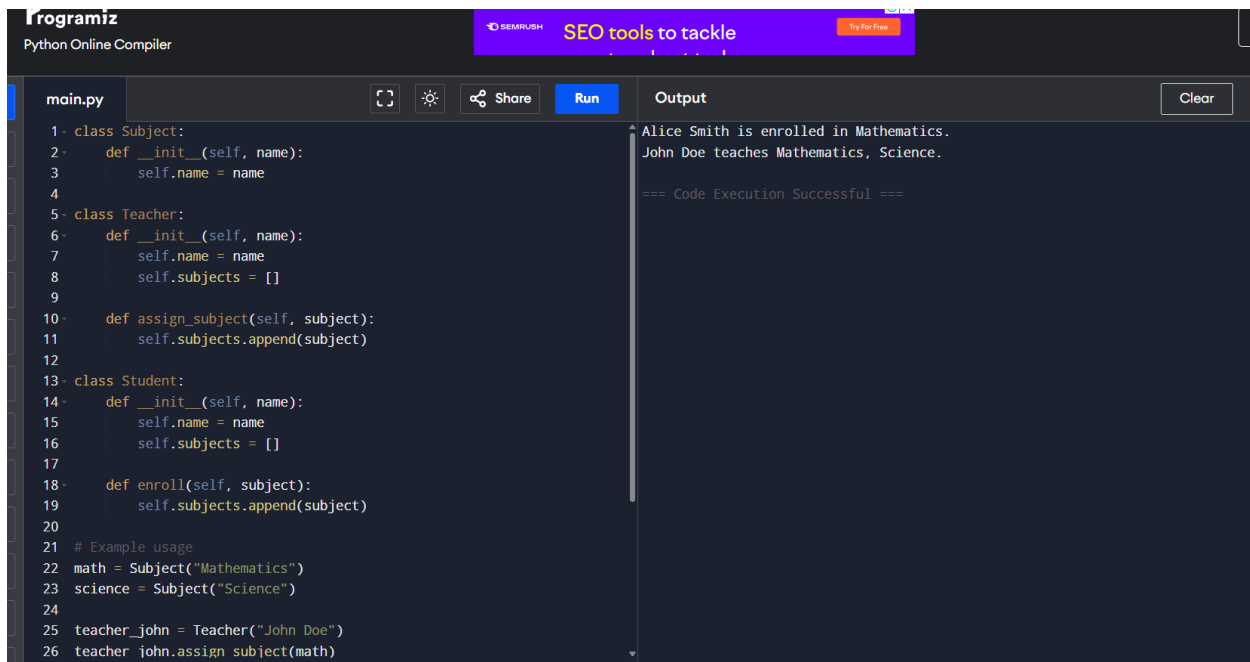
```
science = Subject("Science")
```

```
teacher_john = Teacher("John Doe")
```

```
teacher_john.assign_subject(math)
teacher_john.assign_subject(science)
```

```
student_alice = Student("Alice Smith")
student_alice.enroll(math)
```

```
print(f'{student_alice.name} is enrolled in {', '.join([sub.name for sub in
student_alice.subjects])}.')
print(f'{teacher_john.name} teaches {', '.join([sub.name for sub in teacher_john.subjects])}.')
```



The screenshot shows a web-based Python IDE. The left pane contains a Python script named 'main.py' with the following code:

```
1 class Subject:
2     def __init__(self, name):
3         self.name = name
4
5 class Teacher:
6     def __init__(self, name):
7         self.name = name
8         self.subjects = []
9
10    def assign_subject(self, subject):
11        self.subjects.append(subject)
12
13 class Student:
14     def __init__(self, name):
15         self.name = name
16         self.subjects = []
17
18    def enroll(self, subject):
19        self.subjects.append(subject)
20
21 # Example usage
22 math = Subject("Mathematics")
23 science = Subject("Science")
24
25 teacher_john = Teacher("John Doe")
26 teacher_john.assign_subject(math)
```

The right pane shows the output of the program:

```
Alice Smith is enrolled in Mathematics.
John Doe teaches Mathematics, Science.

=== Code Execution Successful ===
```

20. Write the prolog program for PLANETS DB.

```
% Facts about planets
```

```
planet(mercury, terrestrial, 57.91e6, no).
```

```
planet(venus, terrestrial, 108.2e6, no).
```

```
planet(earth, terrestrial, 149.6e6, [moon]).
```

```
planet(mars, terrestrial, 227.9e6, [phobos, deimos]).
```

```
planet(jupiter, gas_giant, 778.5e6, [io, europa, ganymede, callisto]).
```

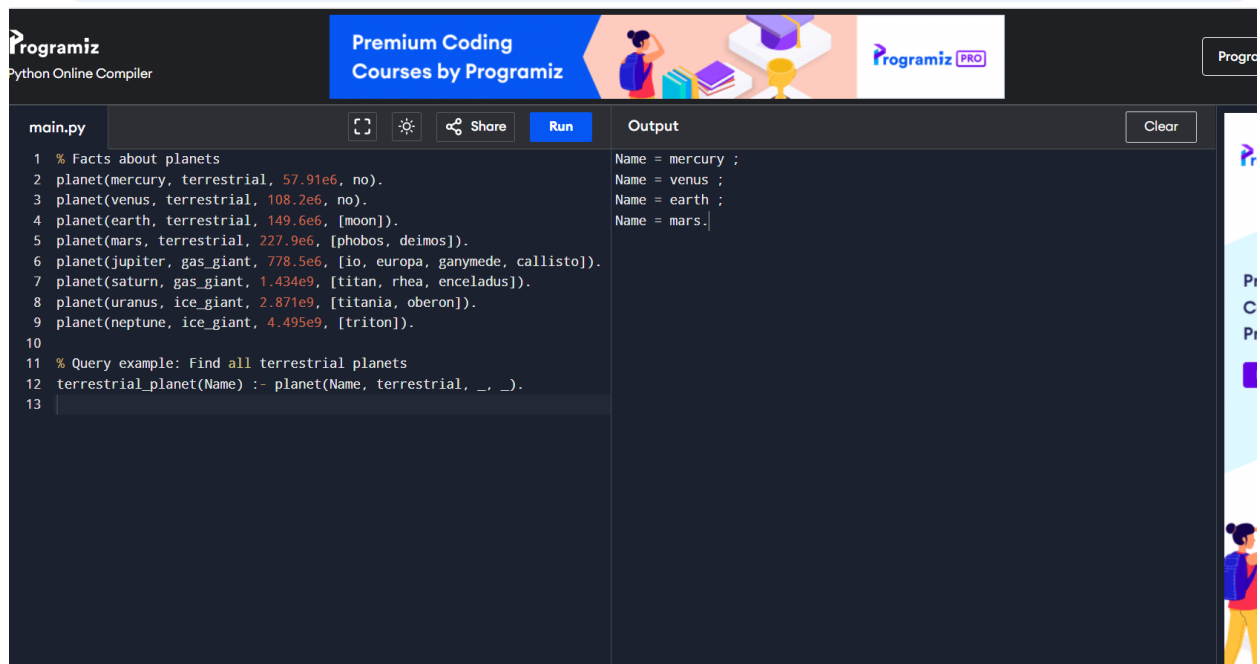
```
planet(saturn, gas_giant, 1.434e9, [titan, rhea, enceladus]).
```

```
planet(uranus, ice_giant, 2.871e9, [titania, oberon]).
```

```
planet(neptune, ice_giant, 4.495e9, [triton]).
```

% Query example: Find all terrestrial planets

```
terrestrial_planet(Name) :- planet(Name, terrestrial, _, _).
```



The screenshot shows the Programiz Python Online Compiler interface. The top header includes the Programiz logo, a banner for "Premium Coding Courses by Programiz", and a "Programiz PRO" badge. The main area is split into two panels: a code editor on the left and an output panel on the right. The code editor contains a Python script with comments and function calls. The output panel shows the results of the script execution.

```
main.py  [Icons]  Share  Run  Clear
```

```
1 % Facts about planets
2 planet(mercury, terrestrial, 57.91e6, no).
3 planet(venus, terrestrial, 108.2e6, no).
4 planet(earth, terrestrial, 149.6e6, [moon]).
5 planet(mars, terrestrial, 227.9e6, [phobos, deimos]).
6 planet(jupiter, gas_giant, 778.5e6, [io, europa, ganymede, callisto]).
7 planet(saturn, gas_giant, 1.434e9, [titan, rhea, enceladus]).
8 planet(uranus, ice_giant, 2.871e9, [titania, oberon]).
9 planet(neptune, ice_giant, 4.495e9, [triton]).
10
11 % Query example: Find all terrestrial planets
12 terrestrial_planet(Name) :- planet(Name, terrestrial, _, _).
13
```

```
Output
Name = mercury ;
Name = venus ;
Name = earth ;
Name = mars.
```